Smoothing the Transition from Agile Software Development to Agile Software Maintenance

Stephen McCalden[1], Mark Tumilty[1], and David Bustard[2(✉)]

[1] Kainos, Belfast BT7 1NT, UK
{S.McCalden,M.Tumilty}@kainos.com
[2] Ulster University, Coleraine BT52 1SA, UK
dw.bustard@ulster.ac.uk

Abstract. Kainos is a software company based in Belfast, Northern Ireland. As well as bespoke development, its work includes service contracts for the maintenance of software created elsewhere. This type of work is challenging because of the knowledge transition involved. The experience reported here is of tackling such projects in a way that integrates with the agile processes of the client. Background on agile practice in Kainos is discussed before focusing on a specific project for the UK Government Cabinet Office.

Keywords: Agile software development · Development-maintenance transition · Scrum · Kanban · Case study

1 Introduction

Kainos [1] is a public limited software company, established in 1986 and based in Belfast, Northern Ireland. It develops information technology solutions for businesses and organizations, particularly in the public, healthcare and financial services sectors. The company also provides consulting and support services. Kainos has offices in the UK, Ireland, Poland and the US, operating across Europe, the Middle East, Africa and North America. It has grown rapidly in recent years, with employee numbers of 260 in 2010, 350 in 2012, and now over 750 in 2015, of whom approximately 490 are engaged in development and 95 in service support (maintenance).

Roughly, three-quarters of the work in the Service Support Department is concerned with software developed by the company. There are several major projects, however, where development took place elsewhere. In such cases, there is a significant challenge in taking on the software in a way that is relatively seamless for the client. The central concern is knowledge acquisition, with the goal being to build an understanding of all aspects of the software without any adverse effect on the service provided in the transition period.

The paper focuses on the situation where a client has had an agile way of working during development and wishes to work with Kainos using the same process during maintenance. The experience of a specific project with the UK Government Cabinet Office is described. This is preceded by background information on agile practices within Kainos.

H. Sharp and T. Hall (Eds.): XP 2016, LNBIP 251, pp. 209–216, 2016.
DOI: 10.1007/978-3-319-33515-5_18

2 Background

The introduction of agile techniques at Kainos started with development. This began in the late 1990s with the introduction of DSDM, but has subsequently been overtaken by the use of Scrum. As noted in surveys carried out in 2010 and 2012 [2], the use of an agile approach in the company was mostly dictated by the requirements of each client. At that time, this did not include the public sector part of the business. Now, however, following a strong government commitment to agile practice in public projects [3], roughly 70 % of the work at Kainos is currently agile-based.

The use of agile techniques in the Service Support Department is a relatively recent innovation, starting in 2013. Like software development, this has largely been client-led. Because of its significant involvement in the public sector, maintenance practices in the company are ITIL-based [4], with ISO20000 IT Service Management accreditation [5] awarded in 2009. As a result, all aspects of the maintenance process are defined in detail, and audited for conformance annually. The process definition includes the role and content of Service Level Agreements (SLAs) and the provision of specific services, such as a Service Desk and the management of incidents and third-party suppliers. The transition process from development to maintenance is also specified and this will be considered further in the next section. Overall, the resulting process has proved very effective, for both Kainos and its clients.

The Service Department at Kainos supports over 80 clients. Each client has an assigned service manager and an engineering team, the composition of which varies according to need. There is also a more senior group of service delivery managers who handle the commercial side of the work, including competition for contracts, contract agreement, and the pricing of new work as it emerges.

Conceptually, the engineers form a single pool of staff, where often each engineer will work with several clients simultaneously. Because of this flexible structure, and a focus on responding quickly to client needs, agile techniques were first introduced through Kanban, supplemented with selected Scrum practices. The approach was based on the three key elements of Kanban identified in [6]:

- *Visualize the workflow*: using a Kanban board (whiteboard/wall), mark out columns showing the left to right stages in handling a client request/incident; split the work into pieces, write each item on a card and put it on the board.
- *Limit work in progress* (WIP): assign explicit limits on how many items may be in progress in each workflow column.
- *Measure the lead/cycle time* (average time to complete one item): optimize the process to make the lead time as small and predictable as possible.

Each Kanban board was set up for the clients associated with a specific client manager. All boards started with the same base process but the associated engineers were encouraged to adjust them as they saw fit, in line with Kanban principles. As with any innovation, this worked best where a ‘champion’ emerged to lead the initiative. Where possible, daily stand-ups were used to review progress with client work and, initially, reflect on the effectiveness of the Kanban approach.

As well as refining the structure of each Kanban board there was also a need to align its content with an existing Kainos Incident Management system (KIM), where clients report issues or make requests for change. KIM held all of the information associated with each work item. The Kanban cards simply recorded KIM references and brief summaries of the tasks involved. Aligning these parallel descriptions required discipline and, for most engineers, felt unsatisfactory. Another difficulty was that some engineers occasionally worked offsite, which meant they couldn't see the board or keep it up-to-date with their own activity. To help address both problems an electronic Kanban system, KanbanFlow [7] was introduced in early 2014 and the physical boards replaced by electronic screens. To retain most of the benefits of the original boards, each screen was dedicated to representing the board it replaced. Since then, the only change has been to switch from KanbanFlow to Trello [8], because of its adoption as the general agile support platform within Kainos.

3 The Transition Challenge

Kainos has experience of all three types of software transition [9]:

- *Self-to-self*, where the transition occurs entirely within the developing organization, continuing with the same process, and largely using the same personnel.
- *Intra-organizational*, where the system is passed from a development team to a separate maintenance team within the same organization.
- *Inter-organizational*, where the system is transferred to an entirely separate organization.

One significant example of self-to-self transition is Evolve [10], its electronic medical records system, which currently has 29 Healthcare Trust clients across 70 hospitals in the UK. As a major product, Evolve has a pool of dedicated staff responsible for its promotion, deployment and support. This includes: (i) a client-facing analysis team who work with each new Trust to identify its specific requirements; (ii) a back-end technical team who handle the implementation and deployment of each instance of Evolve, together with the ongoing enhancement of the base product; and (iii) a support team with the maintenance role of responding to client incidents and their requests for change. All of this work is managed using Scrum.

Roughly 75 % of the projects at Kainos are intra-organizational, involving the transfer of responsibility for systems developed by a Kainos team to its Service Support Department. In some cases, client contracts allow for one or more years of maintenance support on top of initial development. More commonly, however, the company has to win a competitive tendering process to obtain such work.

The remaining 25 % of projects at Kainos are inter-organizational, either in receiving systems developed in other organizations or in passing on systems that it has created. Such arrangements often reflect the preferences of individual clients. For example, some organizations, such as the UK Government, generally develop in-house and then

contract out support responsibility for the resulting systems. Similarly, there are organizations that commission the initial development of systems with the intention of taking responsibility for them after deployment.

When competing for projects, Kainos transition arrangements are made explicit, as part of the contract. These are defined around its ITIL-based support service. In particular, this involves the creation of a Support Handover document for each transition. The document is instantiated from a general template that identifies all of the information that has to be provided by the client. This ranges from basic contact details, through descriptions of the software and associated tools, to a summary of known issues. Checklists are used to ensure that no relevant information is overlooked.

The many activities associated with transition in Kainos fall into three areas:

- *Software transition*, covering the transfer of all software-related artefacts from the development team to the maintenance team, including documentation, test suites, and product backlog, in addition to the software itself.
- *Process transition*, covering the introduction of the way in which the client and maintenance organization will interact.
- *Knowledge transition*, covering the acquisition of knowledge by the maintenance team to the level necessary to take over full responsibility for future changes.

Each of these areas is discussed separately in the sub-sections that follow.

Software Transition. With modern configuration management practices, Kainos find that software transition can usually be completed without difficulty. As well as gaining access to software-related assets, there is a need to examine wider environment arrangements. This involves reviewing:

- *Assets and licenses*, rationalizing if possible.
- *Current infrastructure*, to ensure appropriate environments are in place to resolve software issues and facilitate change; typically, this means ensuring that there are development, test and training environments in place, and that these are consistent with the live environment.
- *Existing environments*, to identify potential security issues, and make recommendations for their resolution, as necessary.
- *3rd party agreements*, if any.

Software transition is largely independent of agile practice, though with agile projects less documentation is expected and a more comprehensive test suite is likely to be in place.

Process Transition. Process transition in Kainos involves the alignment of the working practices of the client with those of the company. This allows for adjustments on both sides to achieve a process that is efficient, effective and satisfactory to all, within the context of the contracted Service Level Agreement. The resulting process covers day-to-day interaction in managing incidents, and higher-level interaction associated with the planning, review and release of new content. This is also the time to introduce support technology, such as the use of an electronic help desk and/or incident management tools.

Process transition is generally straightforward regardless of the practices on either side of the transition. Generally, the maintenance organization adapts to client requirements, though small adjustments on the client side may be necessary. For example, clients accustomed to reporting issues at the end of a sprint cycle, will need to report them immediately in the maintenance phase.

Knowledge Transition. The most difficult aspect of the development-maintenance transition is the acquisition of knowledge by the maintenance team. Training courses can help but there is really no substitute for hands-on experience, preferably with suitable members of the development team available to provide guidance. In Kainos, maintenance teams have found it useful to document their acquired knowledge in an Operations Manual, which is essentially a 'how to' guide for the system they are acquiring.

It is important for clients to be aware of the difficulty of knowledge transfer and allow for it in their planning and costing of maintenance support. The options available are discussed in the next sub-section.

Transition Strategy. The software, process and knowledge transition activities, discussed in the sub-sections above, identify *what* needs to be done during transition, but equally important are the decisions on *where*, *how* and *by whom* these are to be performed. Although, in principle, transition responsibilities could be shared between a development and maintenance team, the work is typically led by the maintenance team, as it is affected most by the success or otherwise of the process.

One major factor influencing the efficiency and effectiveness of transition is the degree of overlap between development and maintenance. With self-to-self and intra-organizational transitions, both occurring within Kainos, there is flexibility in when and how transition is handled. For inter-organizational transitions, however, the process has to be treated formally. There are three situations to consider. The first is where there is no significant overlap between development and maintenance, implying an immediate transfer of system responsibly from one team to another. In such cases, knowledge transition is more difficult, because typically there is little to no communication between the two teams involved, except through documentation. This is more of a problem for agile development projects where less documentation is produced.

Where there is an overlap in transition between development and maintenance, there are two options available:

- *Maintainer-site transition*, where one or more of the development team works on-site with the maintenance team to facilitate transition activities, mostly in a coaching role.
- *Developer-site transition*, where one or more senior members of the maintenance team work on-site with the development team to complete all necessary transition activities; in doing so, the maintenance team members would be involved in production tasks, as an aid to knowledge transition.

Maintainer-site transition has the advantage of occurring at a less pressured time, after deployment, but is typically less satisfactory overall. In particular, development

team members have a weaker role, as they are not driving the transition; also, they are unlikely to be senior members of the team and so may lack a full understanding of all aspects of the system and its support.

Developer-site transition can be a pressured situation if performed around a 'go live' date, which is often the case. A cyclical agile development structure is very helpful here, however, in that it allows the maintenance staff to join a project at the beginning of a sprint, and so be directly involved in its planning and subsequent review. Therefore, while developer-site transition is preferable to having no transition overlap at all, embedding maintenance staff in the development team appears to be the better option. This is the approach described in the next section in a project for the UK Government Cabinet Office.

Transition Example. The UK Government Cabinet Office project is an example of an agile-oriented inter-organizational transition from development to maintenance. It is significant for Kainos in being its first and, so far, only example where a client wished to extend the sprint structure used in development, into maintenance. It is also the first project where the client facilitated transition by supporting service support staff working on-site with the development team. From an agile perspective, the system is additionally significant for the Government Digital Service (GDS) [11] who developed the system, in being their first example of a "*major transactional service delivered all the way to live as an agile project*" [12].

The system, IER (Individual Electoral Registration), provides a single hub through which those eligible to vote in England, Scotland and Wales can register online. This covers 46 million people, across 387 local authorities. The service went live on 10 June 2014. The maintenance contract was awarded to Kainos in the same month, with the transition to maintenance occurring across July and August 2014. Thirty days of Kainos staff time were agreed to support the transition process. Two senior Kainos support engineers (normally based in Belfast) travelled separately to the developer site (in London) for part of each week; one engineer focused on web operations and the other on the remainder of the application (the first named author of this paper). The transition occurred after the 'go live' date, so developers were less pressured, although it did mean that fewer of them were available for consultation.

A full schedule of activity was developed and approved ahead of the on-site transition, indicating the work to be completed each day by each Kainos engineer. As part of their transition schedule, the Kainos engineers worked alongside their counterparts in GDS, assisting with the sprint backlog and working through incidents that occurred.

4 Lessons Learned

The main lessons learned from the Cabinet Office transition project were:

- The timing and general structure of the transition felt close to optimal. Tackling the transition a month after the system went live meant that the development team were available to provide initial support in the crucial first few weeks of release, and then

had time to support transition. There were 20 + lower priority items in the backlog at the go-live point, meaning that there were tasks available to facilitate knowledge transition and keep developers busy when there were no incidents to handle.

- Using developer-site transition proved very effective. With this approach, Service Support in Kainos was able to take on a substantial system, cover all incidents reported (there were very few) and move its development forward — all without any interruption in service. One significant achievement was taking responsibility for the system being rolled out to Scotland, which was delayed until after the independence referendum on 18 September 2014 [13].
- Scheduling transition activities around an agile process is very straightforward. The cyclical nature of the work, and its detailed breakdown in a backlog, meant that it was relatively easy to identify tasks that could be shadowed, and others that could be tackled by the Kainos engineers to build up their experience.
- The GDS development team was very supportive of the transition process, making it fully effective. Greater efficiency may be possible, however, through a tighter collaboration. Specifically, this would involve inserting the transition tasks directly into the sprint backlog of the development team. In that way, transition activities would be covered in sprint planning meetings, daily standups, sprint reviews and sprint retrospectives, with a possibility of reducing the elapsed time of the transition and total effort expended. Further experimentation is needed to evaluate this possibility.
- The Cabinet Office requirement to run support with the same sprint structure as development was largely straightforward. The scale of the work involved meant that a support team could be dedicated to the contract, and work in Scrum cycles. The only difficulty encountered was a need to obtain approval for an exception to ISO 20000 certification to allow for changes to be specified as user stories rather than the usual, more detailed definitions.

Acknowledgements. We are very grateful to the Cabinet Office and GDS for their facilitation of the transition project described in the paper. With their understanding and accommodation, the transition process proved very successful. Thanks also to Kainos for supporting the creation of the paper, especially Tom Gray, the CTO, for his interest and enthusiasm throughout. Finally, of course, we are grateful to our 'shepherd', for his guidance on the structure and content of the paper.

References

1. Kainos. www.kainos.com
2. Bustard, D., Wilkie, G., Greer, D.: Towards optimal software engineering: learning from agile practice. Innovations Syst. Softw. Eng. **9**(3), 191–200 (2013)
3. UK Cabinet Office: Government IT Strategy (2011)
4. Cannon, D.: UK Cabinet Office: Key Element Guide ITIL Service Strategy: Aligned to the 2011 Editions. Stationery Office Books (2012)
5. Cots, S., Casadesús, M.: Exploring the service management standard ISO 20000. Total Qual. Manag. Bus. Excellence **26**(5–6), 515–533 (2015)
6. Kniberg, H., Skarin, M.: Kanban and Scrum-making the most of both. Lulu. com (2010)
7. KanbanFlow. https://kanbanflow.com/
8. Trello. https://trello.com/
9. Khan, A. S.: A Framework for Software System Handover, Doctoral Thesis. Software and Computer Systems, School of Information and Communication Technology (ICT), KTH Royal Institute of Technology, Sweden (2013)
10. Kainos: Evolve Electronic Medical Records Platform. https://www.kainosevolve.com/
11. Government Digital Service. https://www.gov.uk/government/organisations/government-digital-service
12. Government Digital Service Blog: Individual Electoral Registration - changing the way we register to vote (2014). https://gds.blog.gov.uk/2014/06/10/individual-electoral-registration-changing-the-way-we-register-to-vote-2/
13. Wikipedia: Scottish Independence Referendum (2014). https://en.wikipedia.org/wiki/Scottish_independence_referendum,_2014